한국 CAD/CAM 학회 논문집
제 14권 제 3호 2009년 6월 pp. 207-216

재구성이 가능한 다분야통합최적설계 프레임웍의 개발

이장효*, 이세정**

Reconfigurable Multidisciplinary Design Optimization Framework

Janghyo Lee* and Se Jung Lee**

ABSTRACT

Modern engineering design problems involve complexity of disciplinary coupling and difficulty of problem formulation. Multidisciplinary design optimization can overcome the complexity and design optimization software or frameworks can lessen the difficulty. Recently, a growing number of new multidisciplinary design optimization techniques have been proposed. However, each technique has its own pros and cons and it is hard to predict a priori which technique is more efficient than others for a specific problem. In this study, a software system has been developed to directly solve MDO problems with minimal input required. Since the system is based on MATLAB, it can exploit the optimization toolbox which is already developed and proven to be effective and robust. The framework is devised to change an MDO technique to another as the optimization goes on and it is called a reconfigurable MDO framework. Several numerical examples are shown to prove the validity of the reconfiguration idea and its effectiveness.

Key words : Reconfigurability, Multidisciplinary Design Optimization, Framework, Hybrid Approach

1. 서 론

최근의 공학설계문제는 여러 분야에서 다양한 인성을 가진 제품을 효율적이고 효과적으로 설계해야 하는 특성을 지니고 있다. 이러한 문제를 해결하기 위해 다분야통합최적설계(Multidisciplinary Design Optimization: MDO)에 대한 필요성이 증가되었고, MDF(Multidisciplinary Design Feasible)[1], IDF(Individual Discipline Feasible)[2], AAO(All At Once)[2], CO(Collaborative Optimization)[3] 등의 다양한 해석 방법들이 개발되었다. 하지만 실제 설계문제를 해결하는데 위 방법들을 적용하기 위해선 많은 시간과 노력이 요구된다. 모든 방법을 이해하였더라도 기존 MDO 프레임웍을 이용하여 문제를 해결하기 위해서는 각각의 방법에 맞도록 문제를 구성해야 하고, 문제 구성 시 MDO 프레임웍에 대한 전문적인 지식이 요구된다. 또한 사용자가 선택한 방법이 해당 설계문제에 효율적이지 않다면 자신의 설계문제에 적합한 방법을 찾아 문제를 다시 구성해야 하는 불편함을 겪게 된다. 임의의 계산시점에서 다른 방법으로 전환하기 위해서는 변수에 관한 데이터를 저장하였다가 새로운 MDO 방법에 맞도록 문제 구성을 변경한 후 다시 계산을 수행해야 하는 어려움도 있다. 본 연구에서는 이러한 불편함을 최소화 하고 MDO 방법에 관한 전문지식이 없는 사용자들도 프레임웍을 사용할 수 있는 방법을 제안하고자 한다.

본 연구에서는 MDO 문제를 수치적으로 정식화하면 MDO 방법에 관한 추가적인 입력을 요구하지 않고 자동으로 MDF, IDF, AAO, CO의 방법에 따라 문제를 구성해주고, 사용자가 계산 수행 중 다른 MDO 방법으로 변경이 필요할 때에는 추가적인 코딩 없이도 재구성이 가능한 MDO 프레임웍을 개발하였다.

MDO 프레임웍이란 설계 및 해석도구들을 분산컴퓨팅 환경에서 엔지니어가 쉽게 연결하고 이를 자동 실행시킬 수 있는 설계기반을 말한다. 현재 이와 같은 기능을 할 수 있도록 개발된 상용 MDO프레임웍으로는 iSIGHT[4], ModelCenter[5], DAKOTA[6], PIAnO[7]

*비회원, 서울시립대학교 기계정보공학과
**교신저자, 정회원, 서울시립대학교 기계정보공학과
- 논문투고일: 2009. 01. 30
- 논문수정일: 2009. 04. 16
- 심사완료일: 2009. 04. 20

등이 있다. 이들 프레임웍의 특징은 Excel, Matlab외에 ADMAS, ANSYS, ABAQUS와 같은 상용 해석 도구를 통합해 주는 인터페이스를 지원해 준다는 것과 분산된 환경에 존재하는 다양한 해석기를 손쉽게 통합하고 공유할 수 있다는 것이다. 또한 최신의 최적 설계기법을 손쉽게 정의할 수 있는 사용자 인터페이스를 제공하고 있다.

MDO 문제를 재구성 하는 방법에 관한 연구는 현재 활발히 진행 중이다. 문제를 재구성한다는 의미는 사용자가 설계분야에 대한 입력과 출력을 결정하면 분야간 연성변수들 간의 관계를 다양한 형태로 연결시켜 주는 것을 의미한다. Alexandrov는 MDO 문제를 재구성하는 방법으로 REMS[8,9](Reconfigurable Multidisciplinary Synthesis)를 제안하였다. 이 방법은 사용자가 각 분야별로 입력과 출력에 관한 변수를 지정하면 분야간 연성을 파악하여 자동으로 MDA (Multidiscipline Analysis)를 구성하고 여기에 최적화 알고리즘을 결합하여 MDO문제를 풀도록 되어있다.

Martins는 객체지향 방법으로 사용자가 문제구성요소만을 입력하면 MDO방법론에 맞는 형식으로 문제를 구성해 주는 piMDO 프레임웍을 개발하였다[10].

이런 활발한 연구에도 불구하고 아직까지 계산도중 설계 데이터 지점에서 다른 방법으로 문제를 자동 재구성해주는 프레임웍은 개발되지 않았다. 또한 기존의 프레임웍들은 가격이 매우 비싸고 전문가적인 프로그래밍 수준을 요구하기도 하여 능숙하게 사용하기까지 많은 시간과 노력이 요구된다.

따라서 본 연구의 목적은 현재까지 개발된 프레임웍들이 갖추지 못한 설계문제의 재구성 기능과 사용자 편의성에 초점을 맞추어 MDO방법론에 관한 지식이 없더라도 쉽게 사용할 수 있고 간단한 문제 입력만으로 문제구성을 해주면서 계산도중 사용자가 선택한 방법에 따라 자동으로 문제를 재구성해주는 MDO프레임웍을 개발하는데 있다.

이러한 연구는 MDO 문제를 효과적으로 풀기 위한 혼합적 접근법을 결정하는데 많은 도움을 줄 수 있다. 혼합적 접근법이란 설계문제의 연성 정도, 비선형성, 초기값과 경계값의 범위, 해석과 최적화에 걸리는 시간, 분산처리 여부 등 문제의 특성을 고려하여 계산도중 연속적으로 효율적인 방법을 판단하여 선택된 방법으로 계산을 수행해 나가는 방식이다. 즉 MDO문제는 문제의 특성에 따라 한 가지 설계문제를 풀더라도 때에 따라 효과적인 방법이 달라질 수 있기 때문에 하나의 방법으로 푸는 것보다 다양한 방법의 조합으로 푸는 것이 효과적이다. 예를 들어 초기값과 경계값의 범위가 넓은 문제를 풀기 위해 초기에는 수렴성이 좋은 MDF, IDF 등의 방법으로 계산을 수행하다가 어느 정도 수렴이 되었다고 판단이 되었을 경우에는 계산시간이 상대적으로 적게 걸리는 AAO 등으로 결과를 도출할 수 있을 것이다. 현재까지는 계산도중 방법의 전환이 전적으로 설계자의 판단에 따라 이루어지는 수준이다. 하지만 보다 연구가 진행이 되어 다양한 설계경험이 쌓이고 명확한 판단기준이 정립되면 MDO 프레임웍에서 자동으로 효과적인 방법을 판단할 수 있을 것으로 기대된다. 이러한 발전은 앞으로 설계엔지니어들에게 MDO 프레임웍에 대한 접근성을 높임으로서 공학설계문제를 누구나 쉽게 MDO 프레임웍에 적용할 수 있도록 하는데 많은 도움이 될 것이다.

이 논문의 내용은 다음과 같이 구성되어 있다. 2장에서는 MDO 방법론의 내용과 장단점, 방법론간의 연관성과 재구성 방법에 관해 설명하고 3장에서는 MDO 프레임웍 구현에 관련된 요소와 MATLAB에서 어떻게 구현하였는지 설명하였다. 4장에서는 실제 예제를 MDO 프레임웍에 적용하여 결과를 살펴보고 5장에서는 앞으로의 발전방향에 대해 기술하였다.

2. 설계문제 재구성 방법

2.1 MDO 방법론

MDO 설계문제를 해결하기 위한 방법론으로 다양한 방법들이 개발되었지만 이 논문에서는 MDF, IDF, AAO, CO 네 가지 방법에 관해서만 다루기로 하겠다. 각 방법에 대한 개략적인 내용은 다음에 설명하였다. 명확한 설명을 위하여 분야가 두 개인 설계문제를 예를 들었다.

2.1.1 MDF(Multidisciplinary Design Feasible) 방법

MDF 방법은 MDO 문제를 푸는 가장 전통적이고 표준적인 방법으로 간단히 나타내면 Fig. 1과 같다. 상위 시스템에서 설계변수 값을 하위시스템에 주면 하위 시스템에선 관련된 분야들을 동합하는 다분야통합해석을 통해 각 분야의 연성을 고려한 해석을 수행하고 상태변수 값을 되돌려 준다. 즉 MDF의 의미처럼 최적해를 찾아가는 과정에서 나오는 상태변수 값은 다분야를 동시에 만족하는 해이다. MDF 방법은 새로운 보조 변수를 생성하지 않아 설계변수를 증가시키지 않는다는 장점이 있지만 최적 해를 찾는 과정에서 다분야를 만족하는 해를 찾기 위해 많은 계산 시간을 요구한다는 단점이 있다.

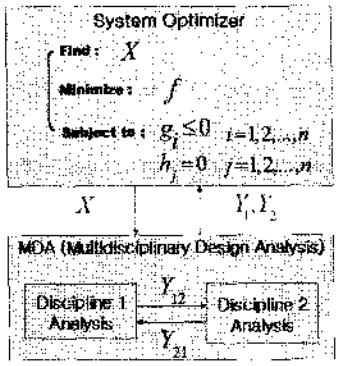

Fig. 1. Variables and data flow of MDF method.

2.1.2 IDF (Individual Discipline Feasible) 방법

IDF방법은 다분야통합해석 분야들 간의 연성을 제거하는 대신 연성변수와 적합성제약조건을 추가하는 방법으로, Fig. 2와 같이 나타낼 수 있다. 상위시스템에서 설계변수와 연성변수 값을 하위시스템에 주면 하위시스템에선 그 값을 가지고 해석을 수행한 후 계산된 연성변수 값을 되돌려 준다. 상위시스템에선 내려주었던 연성변수 값과 되돌려 받은 연성변수 값의 차를 비교하여 적합성제약조건을 만족했는지 판단한다.

IDF방법은 다분야통합해석의 반복계산을 피할 수 있고 각 분야의 해석이 분산환경에서 실행될 수 있는 장점이 있지만 최적화 중간단계에서는 유용한 해를 얻을 수 없다는 것과 MDF방법과 비교하였을 때 연성변수가 설계변수에 추가되고 제약조건도 적합성제약조건이 추가되어 설계변수의 초기값과 경계조건의 범위가 넓을 경우 수렴성이 저하되는 단점이 있다.

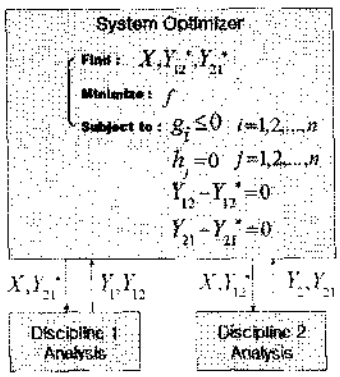

Fig. 2. Variables and data flow of IDF method.

2.1.3 AAO(All At Once) 방법

AAO방법은 하위시스템의 수렴과 상위시스템의 최적화를 동시에 수행하는 특징을 지니는 방법으로 Fig. 3과 같이 나타낼 수 있다. IDF방법과 같이 연성변수와 적합성 제약조건이 추가되는 것에 더불어 각 분야 간의 방정식을 만족하는 해를 찾기 위해 오차(Residual)가 등식제약조건으로 추가된다.

이 방법은 상위시스템의 최적화와 하위시스템의 수렴이 동시에 진행되기 때문에 다른 방법에 비해 시간이 적게 걸리는 장점이 있지만 IDF와 같이 최적화 중간단계에서는 유용한 해를 얻을 수 없다는 것과 상태변수 값이 발산하거나 실수가 아닌 허수가 나오게 되면 최적화의 수렴성도 떨어지는 단점이 있다.

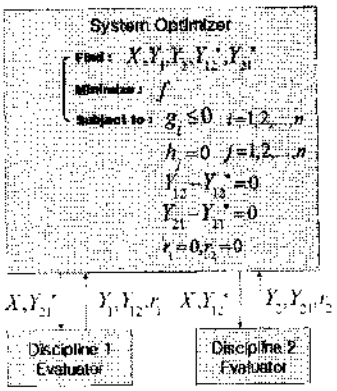

Fig. 3. Variables and data flow of AAO method.

2.1.4 CO(Collaborative Optimization) 방법

CO방법은 다단계 최적화 기법으로 Fig. 4와 같이 나타낼 수 있다. 상위시스템에서 전체의 목적함수를 최소화 시키면서 각 분야의 연성과 관련된 변수들을 결정하고 적합성조건을 만족하는지 확인한다. 하위시스템에서는 상위시스템에서 내려온 공유설계변수와 연성변수, 하위시스템의 설계변수와 상태변수의 최소자승을 목적함수로 하여 그 값을 최소화 하는 최적화

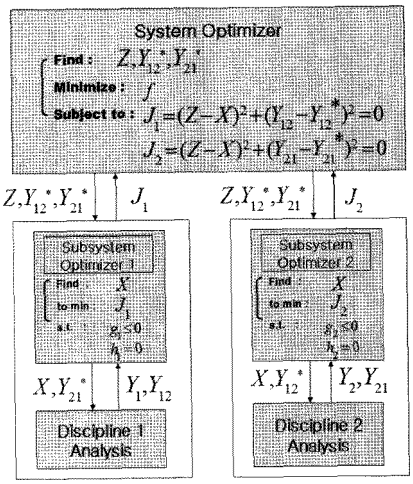

Fig. 4. Variables and data flow of CO method.

를 수행한다.

CO는 각 분야의 자율성이 상조된 기법으로 다단계 최적화 기법을 도입하여 각 분야에 설계변수의 결정권한을 주었고 따라서 분산 최적화가 가능하다. 하지만 기본의 다른 방법에 비해 설계변수와 적합성 제약조건이 늘어나면서 수렴성에 제한이 있고 분야간의 연성이 큰 문제에서는 효율적이지 못한 단점이 있다[11].

2.2 재구성 방법

Fig. 1~4에서 보듯이 설계변수벡터 X와 상태변수벡터 Y_1, Y_2, 연성변수벡터 Y_{12}, Y_{21}, 연성변수 목표 값 Y_{12}^*, Y_{21}^*, 결과값오차(residual)인 r_1, r_2, 다단계 기법에서의 시스템 설계변수 Z, 하위시스템 목적함수 J를 알면 모든 MDO 방법에 맞추어 문제를 구성할 수 있다. 하지만 방법론간의 재구성을 위하여 모든 변수들에 대한 정보를 필요로 한다면 간단한 문제를 해결하는 데에도 수많은 변수를 입력해야 하는 불편함이 따르게 된다. 따라서 본 연구에서는 사용자 입력이 가장 적은 MDF방법을 기준으로 다른 방법으로의 전환을 모색하였다.

2.2.1 최적화 부분의 재구성

MDF방법에서 IDF방법으로 변환할 경우에는 최적화 부분의 설계변수에 연성변수의 목표 값 Y_{12}^*, Y_{21}^*이 추가되며 적합성제약조건 $Y_{12}-Y_{12}^*$, $Y_{21}-Y_{21}^*$이 생성된다. MDF에서 AAO방법으로의 전환은 상태변수 Y_1, Y_2과 연성변수의 목표 값 Y_{12}^*, Y_{21}^*이 최적화 부분의 설계변수에 추가되며 적합성제약조건과 상태방정식의 오차(residual)이 0이 된다는 조건이 추가된다. MDF에서 CO방법으로의 변환은 설계변수 X가 하위시스템 설계변수가 되면서 시스템 변수 Z가 생성되고, 하위시스템 목적함수 J가 등식제약조건으로 추가된다. 이런 추가적으로 요구되는 변수들을 간단히 Fig. 5에 나타내었다. 추가되는 변수들은 오차(residual)을 제외하면 MDF에서 사용되는 변수 X, Y_1, Y_2, Y_{12}, Y_{21}와 연관성이 있다. IDF, AAO, 그 CO에서 모두 사용되는 연성변수 목표 값 Y_{12}^*, Y_{21}^*은 설계자가 입력한 연성변수의 초기값을 최적화 부분에 할당함으로써 해결이 가능하다.

CO방법에서 요구되는 Z와 J는 X와 Y_{12}^*, Y_{21}^*의 조합으로 사용자가 정의한 설계변수와 연성변수의 초기값으로 프레임웍 내에서 자동 정의가 가능하다. 다만 AAO방법의 residual은 각 분야를 m-file로 정의할 때 "evaluation" 부분을 따로 지정을 해줘야만 자동 재구성이 가능한데 이러한 이유는 MDF, IDF, CO 방법은 상태방정식을 만족하는 상태변수를 찾는 해석부분으로 구성되지만 AAO방법은 최적화 부분에 할당된 상태변수에 의한 상태방정식의 계산으로 이루어지는 차이점이 있기 때문이다.

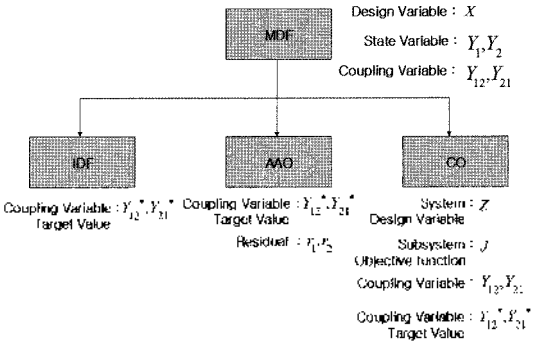

Fig. 5. Additional variables in all MDO methods.

2.2.2 각 분야의 재구성

MDF, IDF, AAO, CO방법에 따라 각 분야의 구성도 달라진다. MDF방법에서는 MDA를 통한 해석을 수행하지만 IDF, AAO, CO에서는 분야간 연성을 제거하여 분야별로 독립적인 계산을 수행하게 된다. 각 방법들간의 자동 재구성을 수행하기 위해선 프레임웍에서 분야별 연성을 찾아 제거해 주거나 또는 MDA로 다시 구성해주는 기능이 필요하다.

이러한 기능을 구현하기 위해 서론에서 언급한 Alexandrov의 REMS 개념을 활용하여 사용자가 입력과 출력을 정의하면 프레임웍에서 연성변수를 찾아 자동으로 MDA를 구성해주도록 하였다. Fig. 6은 두 개의 분야에서 각각 X와 Y_1, Y_3을 입력 받아 Y_3, Y_4와 Y_1, Y_2를 출력하는 문제로 이 문제에서 연성변수는 Y_1, Y_3이 된다. 이것을 MDO 프레임웍에 적용한 모습을 Fig. 7에 나타내었다. 여기서 변수 X는 최적화 부분에서 전달되었다고 가정하고 그림에 표시하지 않았

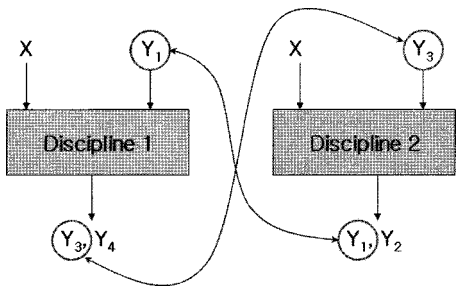

Fig. 6. Input/output of coupling variables between disciplines 1 and 2.

다. 또한 각 분야에서는 상태변수만을 다루기 때문에 초기값으로 연성변수만을 필요로 한다.

이러한 방법으로 MDO 프레임웍 내에서 연성변수를 찾아 MDO방법에 맞도록 데이터를 전달해 줄 수 있다.

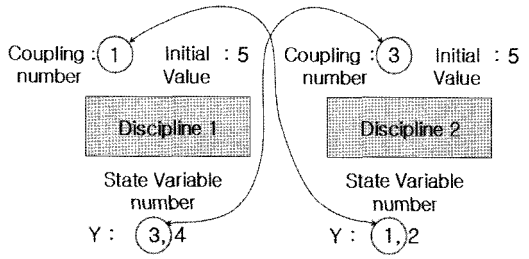

Fig. 7. Implementation of input/output of disciplines 1 and 2 in the MDO framework.

3. MDO 프레임웍의 구현

3.1 MDO 프레임웍 요구조건

MDO 프레임웍의 목적은 MDO 설계문제를 해결하기 위해 다양한 프로그램을 통합하여 실행하는데 있다. 따라서 이러한 기능을 수행하기 위한 요구조건을 세가지 관점에서 요약하면 다음과 같다[12,13].

3.1.1 구조 설계(Architectural Design)

편의성을 고려한 사용자 인터페이스(GUI)를 지원
객체지향기법을 적극 활용
- 프레임웍의 확장성을 위한 CAE 프로그램이나 코드와의 접합성 확보

효율성을 위해 프레임웍이 MDO 분제 해결과정에서 과도한 부담을 주지 말 것
대형 설계문제를 여러 사용자가 협업할 수 있도록 지원

3.1.2 설계문제 정식화(Problem Construction)

고급 프로그래밍(High-level programming) 지원
- 기존에 정식화 되어 있던 문제를 다시 활용 가능
- 상용 해석 코드를 변경하지 않고 적용이 가능
- 다양한 최적화 기법을 제공
- 분산환경에서 실행과정에 대한 디버깅 기능을 지원

3.1.3 설계문제 실행(Problem Execution)
- 계산과정 및 데이터 전송의 자동화가 가능
- 동시에 처리할 수 있는 계산은 병렬처리를 실행
- 분산 환경에서도 실행이 가능
- 실행 중 사용자에 의한 조정이 가능
- 일괄처리 작업이 가능

3.2 MATLAB환경에서의 MDO 프레임웍 구현

본 연구에서는 Matlab을 기반으로 MDO 프레임웍을 구현하였다. Matlab을 선택한 이유는 Optimization Toolbox, Distributed Computing server, GUI 제작을 위한 GUIDE tool 등 프레임웍 구현에 필요한 다양한 기능이 기본적으로 제공되기 때문이다. 또한 많은 공학 엔지니어들이 쉽게 접하는 프로그램이라는 장점도 지니고 있다. 단 본 연구에서는 프레임웍의 외양이나 편의성 등은 최소로 하고, 기능성 위주로 개발하였다.

구조설계부분에서, 사용자 인터페이스는 Matlab의 GUIDE tool을 이용하여 입력을 최소화 할 수 있도록 하였고 숫자, 함수, m-file 이름을 입력하는 것으로 문제를 구성하도록 하였다. CAE프로그램이나 코드와의 접합성은 Matlab의 자체 스크립트 기능으로 여러 실행프로그램과 데이터를 통합할 수 있다. 또한 문제 해결과정에서 프레임웍의 코드 완성도의 영향을 줄이기 위하여 구조를 단순히 하였다.

설계문제 정식화 부분에서 인공한 고급 프로그래밍은 설계문제의 각 분야를 입력할 때 쓰이는 m-file내에서 이루어지도록 하였고 자체 스크립트 기능을 이용하여 기존에 정식화되어 있던 문제를 다시 활용하거나 상용해석 코드를 변경하지 않고 적용이 가능하도록 하였다. 본 연구에서는 MATLAB의 기본 방법인 SQP(Sequential Quadratic Programming) 방법으로 최적화를 수행하도록 하였으며 필요에 따라 다양한 최적화 기법을 통합하여 사용할 수 있을 것이다.

설계문제 실행 부분에서 인공한 계산과정 및 데이터 전송의 자동화는 사용자가 원하는 방법을 선택하였을 때 자동으로 계산이 진행되고 데이터들이 전송되도록 구성하였고 동시에 처리할 수 있는 계산은 Matlab의 Matlabpool 기능을 활용하여 병렬처리 할 수 있다. 하지만 분산 환경에서의 실행, 실행 중 사용자에 의한 조정, 일괄처리 작업등은 이번 프레임웍에서 구현하지 않았다.

또한 이 프레임웍은 MDO 방법의 재구성 측면과 계산 도중 방법을 변환하는 기능에 초점을 맞추어 연구를 진행하였기 때문에 프레임웍에 요구되는 기능 중 상당수 기능이 구현되지 않았다. 하지만 이번에 지원되지 않는 기능들은 향후 추가할 수 있을 것으로 판단된다.

마지막으로 이번 연구에서는 해석분야가 2~4개인 설계문제만 다루었다. 해석분야가 1개인 설계문제는

Matlab에서 제공하는 Optimtool을 이용하여 해결 할 수 있고 5개 이상의 분야를 갖는 설계문제는 단순히 해석분야 확장의 의미를 지닌다고 판단하여 제외하였다. 프레임웍을 사용하는 방법은 4장 적용 사례에서 자세히 설명하였다.

4. 적용 사례

각각의 MDO 방법으로 문제를 푸는 것과 두 가지 이상의 방법을 혼합하여 문제를 푸는 것 중 어떤 방법이 보다 효과적인지 비교하기 위해 세가지 예제를 적용해 보았다. 문제1은 해석분야 2개, 문제2는 해석분야 3개, 문제3은 해석분야 4개로 구성하여 프레임웍에 적용하였다. 문제 풀이 방법도 문제1은 각각의 방법으로 풀은 것과 두 가지 방법을 혼합하여 풀은 결과를 비교하였고 문제2는 정해진 초기값 범위에서 임의의 값을 선택하여 50번의 계산을 수행하였을 때 어떤 방법이 잘 수렴하는지를 비교하였다. 문제3은 세가지 방법을 다양한 순서로 혼합하여 계산해 보고 효율성과 정확성을 비교하였다.

4.1 문제 1

NASA MDOB의 Heart-dipole Problem과 비슷한 형태로 기존문제의 비선형 정도를 줄여서 수렴성을 향상시킨 문제이며 저자가 임의로 만든 예제문제이다.

4.1.1 문제 정식화

이 문제를 정식화 하면 식 (1)과 같다.

$$
\begin{aligned}
&\text{Find} \quad x_1, x_2, x_3, x_4 \\
&\text{Minimize} \quad f_5+f_6+f_7+f_8 \\
&\text{Subject to} \quad f_i > -0,\ i = 5, 6, 7, 8 \\
&\qquad \text{Where } f_5 = y_1y_3 + x_1^2 - 2 \\
&\qquad\qquad f_6 = y_2x_1 + x_2^2 - 2 \\
&\qquad\qquad f_7 = y_3y_4 + x_3^2 - 2 \\
&\qquad\qquad f_8 = y_4y_1 + x_4^2 - 2
\end{aligned} \tag{1}
$$

Discipline 1

$$
\begin{aligned}
f_1 &= y_1^2 + y_2 - 2x_1 = 0 \\
f_2 &= y_2^2 + y_3 - 2x_2 = 0
\end{aligned}
$$

Discipline 2

$$
\begin{aligned}
f_3 &= y_3^2 + y_4 - 2x_3 = 0 \\
f_4 &= y_4^2 + y_1 - 2x_4 = 0
\end{aligned}
$$

4.1.2 문제 구성

Fig. 9는 개발한 MDO 프레임웍에서 문제1의 데이터를 입력한 모습이다. 목적함수의 정의는 식 (1)의 f_5, f_6, f_7, f_8의 합으로 이 예제에서는 각각의 함수를 m-file로 만들었고 이렇게 정의된 f_5, f_6, f_7, f_8을 직접 사용할 수 있다. 다음으로 설계변수의 개수를 4로 입력하면 Fig. 10과 같이 4개의 초기값을 입력할 수 있는 창이 뜬다. 등식, 부등식 제약조건에 관해서도 개수를 입력하면 Fig. 10과 같이 그 수만큼 제약조건을 입력할 수 있는 창이 뜬다. 목적함수처럼 m-file로 정의된 함수를 입력하거나 또는 수식을 직접 입력할 수도 있다. 경계값은 설계변수 개수만큼 입력해주고 해석분야의 개수를 입력하면 Fig. 11과 같이 연성변수, 연성

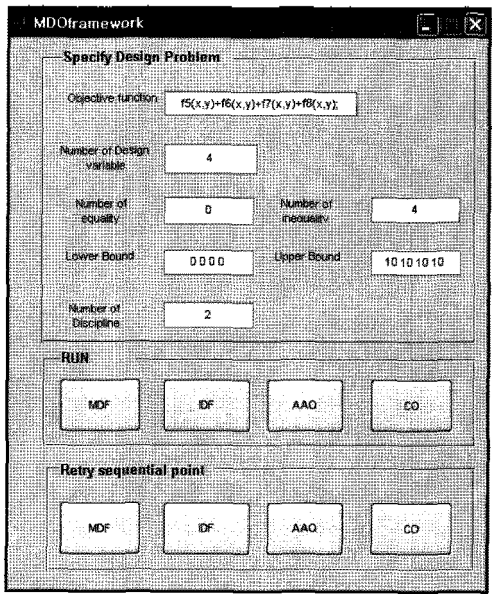

Fig. 8. Problem 1 input: problem definition and MDO methods.

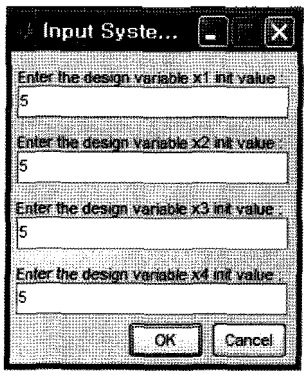

Fig. 9. Problem 1 input: initial values of design variables.

변수의 초기값, 사용자 정의 m-file, 해석분야의 출력변수를 지정할 수 있다.

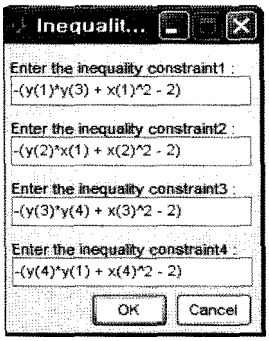

Fig. 10. Problem 1 input: inequality constraints.

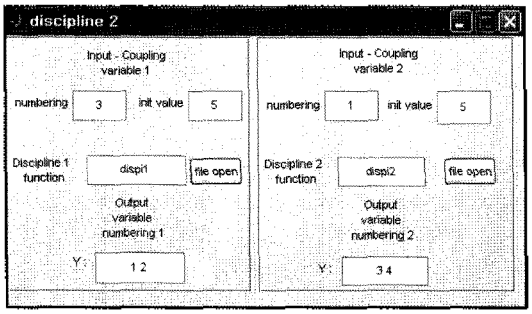

Fig. 11. Problem 1 input: two discipline definitions.

4.1.3 적용 결과

Table 1은 한번의 문제 구성 후 각각의 방법론으로 풀었을 때의 결과를 정리한 것이다. MDF방법을 기준으로 각 분야별 입력과 출력변수, 연성변수를 정의하면 사용자는 추가적인 입력 없이도 MDO 방법에 따라 원하는 방법을 선택하여 문제를 풀 수 있다. 문제 1의 경우 AAO, IDF, MDF, CO방법 순서로 효율성 측면이 뛰어난 것으로 판단된다. 두 가지 방법을 혼합하여 풀었을 경우에는 IDF&AAO, MDF&AAO, MDF&IDF 순서로 효율성이 우수하며 한가지 방법만으로 풀었을 경우와 비교해 평균적으로 적은 함수호출횟수를 보였다.

즉 총 함수호출횟수가 적을수록 문제를 푸는데 요구되는 시간이 짧아지는데 따라서 이 문제의 경우 어떤 방법에서 시작하였더라도 AAO 방법으로 전환하여 문제를 푸는 것이 효율적으로 시간을 단축할 수 있다. 각각의 방법과 혼합된 방법의 총 함수 호출횟수를 Fig. 12에 나타내었다.

Table 1. Comparison of various methods for Problem 1

Methods Used	Final Objective	Constraint Violation	Function Evaluations
MDF	6.15×10^{-4}	4.50×10^{-4}	1,730
IDF	3.9×10^{-3}	2.41×10^{-3}	243
AAO	1.56×10^{-11}	1.56×10^{-11}	167
CO	7.65×10^{-3}	7.10×10^{-3}	67,538
MDF&IDF	4.92×10^{-6}	0	244
MDF&AAO	1.11×10^{-11}	3.31×10^{-15}	192
IDF&AAO	1.40×10^{-16}	6.95×10^{-14}	147

4.2 문제2

이 문제는 2개의 분야로 이루어져 있는 문제이지만 예제의 다양성을 위하여 3개의 분야로 정식화 하였다.

4.2.1 문제 정식화

GL 예제문제를 3개의 분야로 정식화 하면 식 (2)와 같다.

$$
\begin{aligned}
&\text{Find} \quad x_1, x_2, x_3\\
&\text{Minimize} \quad y_2 - y_5\\
&\text{Subject to } 0\le y_3,\ 0<y_4,\ 0\le x_1\le 7,\ 0\le x_2\le 7,\ 0\le x_3\le 7\\
&\text{Discipline 1}\\
&\qquad y_1 = x_1^2 + x_2 + x_3 - 4 \quad 0.2y_1\\
&\qquad y_2 = y_1 - (x_2 - 2)^2\\
&\text{Discipline 2} \qquad\qquad (2)\\
&\qquad y_3 \quad \frac{y_1}{8} \quad 1\\
&\qquad y_4 \quad x_1 \quad x_2 - 2 - \sqrt{y_1}\\
&\text{Discipline 3}\\
&\qquad y_5 = x_3 \quad 2 \mid e^{y_4}\\
&\qquad y_6 = 1 - \frac{y_4}{10}
\end{aligned}
$$

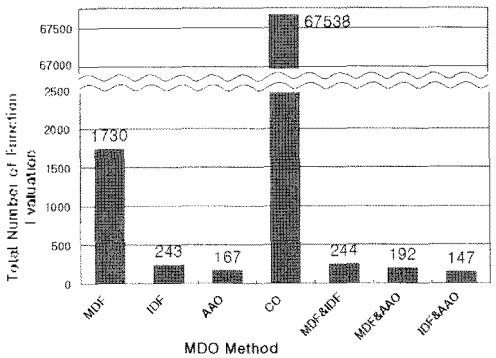

Fig. 12. Comparison of single approaches and hybrid approaches for Problem 1.

4.2.3 적용 결과

이번 문제는 효율성을 측정하였던 문제1과는 달리 다양한 초기값 범위에서 수렴 정도를 파악하여 강건성을 평가하였다. 설계초기값을 경계값 범위 내에서 임의의 값으로 설정한 다음 각각의 방법으로 50회씩 문제를 풀었으며, 그 결과는 Table 2, Fig. 13에 나타내었다. Table 2의 수치는 50회의 결과 중 해로 수렴한 경우만 표시하였고 MDF에서 IDF, AAO, CO방법으로의 전환 시기는 임의로 최적화 부분에서 세 번의 계산을 마친 후로 정하였다.

결과를 살펴보면 MDF와 IDF는 경계값 범위내의 어떤 설계 값에서도 모두 수렴하였고 AAO는 58%, CO는 42%의 수렴성을 보였다. 3번째 iteration에서 일시 중단하고 다른 방법으로 전환하는 방식으로 두 가지 방법을 혼합하여 계산한 결과 MDF&IDF는 모두 수렴하였고 MDF&AAO는 92%, MDF&CO는 78%의 수렴성을 보였다. 결과에서 알 수 있듯이 AAO와 CO방법의 경우 MDF와 IDF를 혼합하여 풀 때 강건성이 향상되었다. 즉 문제2에서 해를 모르고 있는 경우라면 MDF나 IDF방법으로 풀어야 쉽게 해를 찾아 갈 수 있고, 어느 정도 해에 수렴했다고 판단되면 평균 함수 호출횟수가 적은 IDF나 AAO방법으로 전환하여 문제를 해결하는데 소요되는 시간을 단축시킬 수 있다. 방법을 전환하기 위한 구체적인 기준에 관한 연구가 진행된다면 가장 효율적인 시기에 방법의 전환이 이루어 질 것으로 예상된다.

Table 2. Comparison of various methods for Problem 2

Methods Used	Convergence Rate (%)	Final Objective	Function Evaluations
MDF	100	8.0029	703
IDF	100	8.0029	68
AAO	58	8.0029	117
CO	42	8.0031	68,197
MDF&IDF	100	8.0029	174
MDF&AAO	92	8.0029	243
MDF&CO	78	8.003	6,705

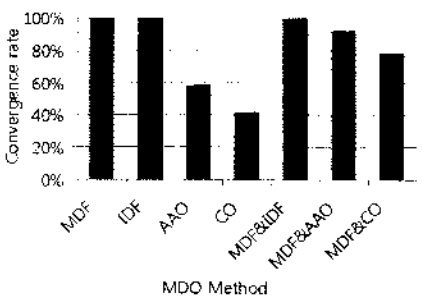

Fig. 13. Comparison of single approaches and hybrid approaches for Problem 2.

4.3 문제 3

이 문제는 NASA의 인공심장 문제로 MDO 테스트 예제이다. 8개의 비선형 방정식의 해를 구하는 문제이지만 본 연구에서는 4개의 분야를 가진 설계문제로 정식화 하였다.

4.3.1 문제 정식화

문제3을 4개의 분야로 정식화 하면 식 (3)과 같다.

$$
\begin{aligned}
&\text{Find} && x_1, x_2, x_3, x_4\\
&\text{Minimize} && f_5+f_6+f_7+f_8\\
&\text{Subject to} && f_i >= 0,\ i = 5, 6, 7, 8\\
&\text{Where } f_5 = x_1(y_3^2 - x_4^2) - 2y_2y_3x_4 \\
&\qquad + y_1(x_3^2 - y_4^2) - 2x_2x_3y_4 - 1\\
&\qquad f_6 = y_2(y_3^2 - x_4^2) + 2x_1y_3x_4 \\
&\qquad + x_2(x_3^2 - y_4^2) + 2y_2x_3y_4 - 1\\
&\qquad f_7 = x_1y_3(y_3^2 - 3x_4^2) + y_2x_3(y_4^2 - 3x_3^2) \\
&\qquad + y_1x_3(x_3^2 - 3y_4^2) \qquad (3)\\
&\qquad + x_2y_4(y_4^2 - 3x_3^2) - 1\\
&\qquad f_8 = y_2y_3(y_3^2 - 3x_4^2) - x_1x_4(x_4^2 - 3y_3^2) \\
&\qquad + x_2x_3(x_3^2 - 3y_4^2) \\
&\qquad - y_1y_4(y_4^2 - 3x_3^2) - 1
\end{aligned}
$$

Discipline 1

$$f_1 = x_1 + x_2 - 1 = 0$$

Discipline 2

$$f_2 = x_3 + x_4 - 1 = 0$$

Discipline 3

$$f_3 = x_5x_1 + x_6x_2 - x_7x_3 - x_8x_4 - 1 = 0$$

Discipline 4

$$f_4 = x_7x_1 + x_8x_2 + x_5x_3 + x_7x_4 - 1 = 0$$

4.3.2 적용 결과

문제 3은 세가지 방법을 혼합하여 계산하고 총 함수호출횟수와 정확성을 비교해 보았다. 세 가지 방법을 순서만 바꾸어서 계산하였지만 계산 결과는 Table 3과 같이 모두 다르게 나타났다. 그 중에서 초기에 AAO, 중기에 MDF, 후기에 IDF방법을 이용하여 해

를 구했을 때 정확도도 가장 높으면서 총 함수호출횟수도 가장 적었다.

계산과정을 초기, 중기, 후기로 나누어 계산해본 이유는 정확한 해를 모르는 경우에 해의 수렴판단 여부를 짐작하기 어려울 수 있으므로 다양한 방법으로의 전환을 실험해 본 것이다. 하지만 세 가지 이상의 방법을 혼합하여 계산하는 것은 경우에 따라 비효율적인 방법이 될 수도 있다. 문제 해결과정은 수렴과정과 정확한 해를 찾는 두 과정으로 분류할 수 있는데 세 가지 방법을 혼합하였을 경우 하나의 방법이 효율적인 계산을 방해하는 장애물이 될 수도 있기 때문이다. 문제3의 경우 각각의 과정에 효율적인 방법을 선택하여 계산하는 것이 가장 빠르고 정확한 해를 얻을 수 있었다.

Table 3. Comparison of various methods for Problem 3

Methods Used	Initial Objective	Final Objective	Function Evaluations
MDF&IDF&AAO	-776	7.305×10^{-7}	864
MDF&AAO&IDF	-776	8.821×10^{-9}	516
IDF&MDF&AAO	-776	4.167×10^{-7}	723
IDF&AAO&MDF	-776	1.344×10^{-6}	1,392
AAO&MDF&IDF	-776	5.717×10^{-10}	493
AAO&IDF&MDF	-776	2.476×10^{-8}	995

5. 결 론

사용자가 MDO 문제의 목적함수, 설계변수의 개수와 초기값, 경계 값, 제약조건, 각 해석분야들의 입력과 출력을 지정해 주었을 때 MDO 방법에 따른 문제의 구성과 계산도중 문제의 자동 재구성 가능성을 연구하였다. MDF방법을 기준으로 사용자에게서 최소한의 입력을 받아 다른 방법으로의 문제구성과 계산도중 방법의 전환은 모두 성공적으로 이루어 졌다.

이러한 문제 재구성이 가능한 MDO 프레임웍은 사용자가 MDO 방법을 모르더라도 사용할 수 있기 때문에 아직까지 비싸고 사용하기 어려운 프로그램으로 인식되어온 MDO 프레임웍을 사용하기 간편하고 편리한 프로그램으로 인식시켜줄 것이다. 또한 엔지니어들에게 MDO 문제에 대한 접근 성을 높이고, 다양한 설계분야에 MDO가 적용되는 효과를 볼 수 있을 것으로 기대된다.

또한 문제해결 과정 중간에 방법을 변환할 수 있게 함으로써 한 가지 방법만으로 MDO문제를 푸는 것이 아닌 두 가지 이상의 방법을 혼합하여 계산을 수행할 수 있게 되었다. 두 가지 이상의 방법을 혼합하여 계산할 경우 방법을 전환하는 판단기준에 관한 연구가 진행되어 있지 않아 현재까진 전적으로 사용자의 판단에 의해 변환이 이루어지고 있다. 하지만 두 가지 이상의 방법을 혼합하여 문제를 풀 경우, 문제1, 문제2, 문제3을 풀이본 결과 한가지 방법으로 문제를 풀 때보다 효율적인 문제 풀이가 가능하였다. 다양한 MDO문제를 풀이보면 상황에 따라 효율적인 방법을 선택할 수 있는 데이터가 쌓일 것이다. 그 데이터를 바탕으로 문제 특성과 계산과정에 따른 구체적인 방법 전환 판단기준을 수립하여 프레임웍에 적용하게 된다면 사용자의 판단이 개입되지 않아도 가장 효율적인 방법으로 문제를 해결해 줄 수 있는 프레임웍이 만들어 질 것이다.

감사의 글

이 논문은 2008년도 서울시립대학교 학술연구 조성비에 의하여 연구되었음.

기호설명

f : 최적화 알고리즘의 목적함수
g_i : 최적화 알고리즘의 부등식 제약조건
h_j : 최적화 알고리즘의 등식 제약조건
X : 설계변수
Y_i : 상태변수
Y_{ij} : 연성변수
r_i : 분야의 오차
Z : 시스템 설계변수
J : 하위시스템 목적함수

참고문헌

1. Cramer, E. J., Frank, P. D., Lewis, R. M. and Shubin, G. R., "Problem Formulation for Multidisciplinary Optimization", *SIAM J. on Optimization*, Vol. 4, No. 4, pp. 754-776, 1994.
2. Tedford, N. P. and Martins, J. R. R. A., "On the Common Structure of MDO Problems : A Comparison of Architectures", AIAA Paper 2006-7080, Sept. 2006.
3. Braun, R. D. and Kroo, I. M., "Development and Application of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment", in Multidisciplinary Design Optimization: State of the Art, N. M. Alexandrov and M. Y. Hussaini, eds.,

SIAM, pp. 98-116, 1997.
4. http://www.engineous.com
5. http://www.phoenix-int.com
6. Eldred, M. S., Giunta, A. A. and Hart, W. E., "DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis", Version 3.3 Reference Manual, Sandia National Lab., 2004.
7. http://www.iframax.com
8. Alexandrov, N. M. and Lewis, R. M., "Reconfigurability in MDO Problem Synthesis, Part 1", AIAA Paper 2004-4307, Aug. 2004.
9. Alexandrov, N. M. and Lewis, R. M., "Reconfigurability in MDO Problem Synthesis, Part 2", AIAA Paper 2004-4308, Aug. 2004.
10. Marriage, C. J. and Martins, J. R. R. A., "Reconfigurable Semi-analytic Sensitivity Methods and MDO Architectures within the πMDO Framework", AIAA Paper, 2008-5956, Sept, 2008.
11. 양영순, 정헌승, "다분야통합 설계 최적화 문제의 정식화 기법에 대한 고찰 Part 2: MDO 정식화 기법의 정류와 특성", 대한조선학회, 제37권, 제3호, pp. 50-52, 2000.
12. Salas, A. and Townsend, J., "Framework Requirements for Multidisciplinary Application Development", 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Mo, Sept., 1998.
13. 주민식, 이세정, 최동훈, "다분야통합최적설계를 지원하는 분산환경 기반의 설계 프레임워크 개발", 한국 CAD/CAM학회 논문집, Vol. 9, No. 2, pp. 143-150, 2005.

이 장 효

2007년 서울시립대학교 기계정보공학과 학사
현재 서울시립대학교 기계정보공학과 석사과정
관심분야: Design Optimization, Framework

이 세 정

1980년 서울대학교 기계공학
1982년 한국과학기술원 기계공학
1989년 펜실베니아주립대학 기계공학
현재 서울시립대학교 기계정보공학과 교수
관심분야: Design Automation, Optimization